Problem 5.

We want to use a quadrature rule to compute the integral

$$\int_0^\infty f(x) \exp(-x) \, \mathrm{d}x. \tag{1}$$

The presence of an infinite integration interval makes it impossible to directly apply a standard quadrature rule. In applications, this can be addressed by truncating the interval of (1) to $[0, T]$ for some large $T > 0$, however, there are more elegant and usually more accurate methods such as the Gauss-Laguerre quadrature rule.

The basis of the Gauss-Laguerre quadrature are the Laguerre polynomials L_n and L_{n+1} defined below. We use the roots $r_i, i = 1, 2, \ldots, n$, of L_n as quadrature nodes and define the weights as

$$w_i = \frac{r_i}{(n+1)^2 L_{n+1}(r_i)^2},$$

allowing us to write the overall quadrature rule as

$$Q_n[f] = \sum_{i=1}^{n} w_i f(x_i). \tag{2}$$

(a) The Laguerre polynomials $L_0, L_1, \ldots$ are given by the three term recurrence

$$L_0(x) = 1, \quad L_1(x) = 1 - x, \quad (n+1)L_{n+1}(x) = (2n + 1 - x)L_n(x) - nL_{n-1}(x). \tag{3}$$

Implement a Python function `laguerre(degree)` which takes the desired degree of the Laguerre polynomial and returns a NumPy `Polynomial` object equal to $L_{\texttt{degree}}$ using (3) (you can import this class from `np.polynomial.polynomial.Polynomial` or use the alias `poly` in the Jupyter notebook we provided on Moodle). Use recursion for this implementation.

Hint: When implementing multiplications like $q(x) = (1 - x) * p(x)$ in Python you need to use a separate `Polynomial` object for the $1 - x$ factor.

(b) Internally, SciPy's `roots_laguerre` function uses an eigenvalue computation to find the roots of $L_n(x)$. In analogy to Theorem 2.12 and Exercise 2 of Series 5 we can use the recurrence (3) to find a tridiagonal matrix A such that

$$A = \begin{pmatrix} a_1 & b_1 & & & & \\ b_1 & a_2 & b_2 & & & \\ & b_2 & a_3 & b_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_{n-2} & a_{n-1} & b_{n-1} \\ & & & & b_{n-1} & a_n \end{pmatrix}$$

with the sequences $(a_1, a_2, \ldots, a_n) = (1, 3, \ldots, 2(n-1) + 1)$ and $(b_1, b_2, \ldots, b_{n-1}) = (-1, -2, \ldots, -n + 1)$.

Show that any root λ of L_n is an eigenvalue of A. Implement a Python function `roots(degree)` that computes the roots of L_n.

Hint: You can use SciPy's `linalg.eigvals_banded` function. In the Jupyter notebook on Moodle you can directly run `eigvals_banded`. Make sure your function returns the proper eigenvalue for degree 1.

(c) Use the roots $r_1, r_2, \dots, r_n$ you found in (b) and verify their quality by comparing the maximum absolute value of $L_n(r_i)$ to 0. Repeat this verification for the first $M > 0$ Laguerre polynomials and plot the results in an appropriate plot. Make sure you that M is not too large.

(d) Implement a Python function `gauss_laguerre(f, num_points)` that computes the Gauss-Laguerre quadrature rule Q_n given in (2). Compute the integral for the function $f(x) = \sin(x)$. Compute the error with respect to the exact integral $\int_0^\infty \sin(x)\exp(-x)\,\mathrm{d}x = 0.5$ for your implementation of the quadrature points and weights, and that of SciPy using `roots_laguerre`. Use $n \in$ `np.arange(1, 25)`. Plot the errors in a plot of your choice. Measure the computational time both quadratures require with the `time` function and plot the elapsed times in a semi-logarithmic plot; use the function `plt.semilogy`/ `matplotlib.pyplot.semilogy`.

The following part is for your understanding and will not be graded: Why is your implementation so much slower than that of SciPy? If you are really interested, you can have a look at the Cython library (cython.org).

Solution.

(a – c, e) You can find our implementation and explanations in the Jupyter notebook we provided on Moodle.

(d) We start from the recursion

$$(n+1)L_{n+1}(x) = (2n+1-x)L_n(x) - nL_{n-1}(x).$$

By defining the vector $\mathbb{L}(x) = [L_0(x), L_1(x), \dots, L_{n-1}(x)]^\perp$ we can see that the following matrix equation holds for all values of x:

$$\begin{pmatrix} 1 & -1 & & & \\ -1 & 3 & -2 & & \\ & & \ddots & & \\ & & -n+2 & 2(n-2)+1 & -n+1 \\ & & & -n+1 & 2(n-1)+1 \end{pmatrix} \mathbb{L}(x) - x\mathbb{L}(x) = n \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ L_n(x) \end{pmatrix}$$

Now, let λ be a root of $L_n(x)$ and define A as the matrix in the previous equation. This means that we have

$$0 = A\mathbb{L}(\lambda) - \lambda\mathbb{L}(\lambda) = (A - \lambda\,\mathrm{id})\mathbb{L}(\lambda).$$

Generally, $\mathbb{L}(\lambda) \neq 0$ and thus $\det(A - \lambda\,\mathrm{id}) = 0$ has to be true. Therefore, we have shown that λ is a root of $\det(A - \lambda\mathrm{id})$, concluding the proof.

You can find our implementation in the Jupyter notebook we provided on Moodle.